L Number	Hits	Search Text	DB	Time stamp
1	33	(((legacy) near4 (C++ or object)) and 717/.ccls.) and inherit$4	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/18 10:30
2	72	"L33" and cast$3	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/18 10:31
3	7	((((legacy) near4 (C++ or object)) and 717/.ccls.) and inherit$4) and cast$3	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/18 11:15
4	50	"C++ wrapper"	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/18 11:16
5	12	map$4 same inherit$4 same (C or legacy or non$1object) same (C++ or object$1oriented)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/18 11:43
6	69	(link$4 or access$5 or retriev$4 or map$4) same inherit$4 same (C or legacy or non$1object) same (C++ or object$1oriented)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/18 12:18
7	57	((link$4 or access$5 or retriev$4 or map$4) same inherit$4 same (C or legacy or non$1object) same (C++ or object$1oriented)) not (map$4 same inherit$4 same (C or legacy or non$1object) same (C++ or object$1oriented))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/18 11:44
8	792	(link$4 or access$5 or retriev$4 or map$4) same inherit$4 same (C or legacy or non$1object)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/18 14:11
9	2	((link$4 or access$5 or retriev$4 or map$4) same inherit$4 same (C or legacy or non$1object)) and static near1 cast$3	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/18 14:10
10	27	(link$4 or access$5 or retriev$4 or map$4) same inherit$4 same (legacy or non$1object)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/18 14:13
11	6	(link$4 or access$5 or retriev$4 or map$4) same inherit$4 near3 (legacy or non$1object)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/18 14:51
12	40	(link$4 or access$5 or retriev$4 or map$4) and inherit$4 near3 (legacy or non$1object)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/18 14:52
-	305	map$4 same (C or non$1object) same (C++ or object$1oriented)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/18 11:38

-	30	(map$4 same (C or non$1object) same (C++ or object$1oriented)) and (no or zero$5 or empty)near3 memory	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/13 17:22
-	0	((map$4 same (C or non$1object) same (C++ or object$1oriented)) and (no or zero$5 or empty)near3 memory) and cad	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/13 17:17
-	30	((map$4 or cast$3) same (C or non$1object) near3 (C++ or object$1oriented)) and (no or zero$5 or empty)near3 memory	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/13 17:22
-	207	(map$4 or cast$3) same (C or non$1object) near3 (C++ or object$1oriented)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/13 17:27
-	8	((map$4 or cast$3) same (C or non$1object) near3 (C++ or object$1oriented)) and cad	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/13 17:26
-	24	(map$4 or cast$3) near4 (C or non$1object) near3 (C++ or object$1oriented)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/13 17:31
-	2	(map$4 or cast$3) near4 (C or non$1object) near3 (C++ or object$1oriented) and cad	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/13 17:32
-	2	(map$4 or cast$3) near4 (C or non$1object) near4 (C++ or object$1oriented) and cad	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/13 17:33
-	24	(map$4 or cast$3) near4 (C or non$1object) near4 (C++ or object$1oriented)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/13 17:33
-	2562	(map$4 or inherit$5) same (C or non$1object or legacy) same (C++ or object)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 09:56
-	236	((map$4 or inherit$5) same (C or non$1object or legacy) same (C++ or object)) and cast$4	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 09:51
-	735	(map$4 or inherit$5) same (C or non$1object or legacy) near4 (C++ or object)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 09:57
-	971	(non$1object or legacy) near4 (C++ or object)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 10:10
-	5	((non$1object or legacy) near4 (C++ or object)) and (map$4 or inherit$5) near6 (non$1object or legacy) near4 (C++ or (object$1oriented))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 09:58

-	103	((non$1object or legacy) near4 (C++ or object)) and 717/.ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 14:44
-	655	(non$1object) near4 (C++ or object)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 10:24
-	52	((non$1object) near4 (C++ or object)) and 717/.ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 10:10
-	1120	(map$4 or retriev$4 or access$5 or inherit$5) same (C) near5 (C++ or (object$1oriented))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 10:25
-	1120	(map$4 or retriev$4 or access$5 or inherit$5) same (C near5 (C++ or (object$1oriented)))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 10:26
-	131	(map$4 or retriev$4 or access$5 or inherit$5) near5 (C near5 (C++ or (object$1oriented)))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 10:26
-	1422	((C or procedure or non$1object or legacy) near4 (C++ or object)) and 717/.ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 19:56
-	8239	(map$3 or link$4 or inherit$5 or refer$5 or retriev$5 or access$5)same ((C or procedure or non$1object or legacy) near4 (C++ or object))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 15:02
-	725	((map$3 or link$4 or inherit$5 or refer$5 or retriev$5 or access$5)same ((C or procedure or non$1object or legacy) near4 (C++ or object))) and 717/.ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 14:53
-	138	(((map$3 or link$4 or inherit$5 or refer$5 or retriev$5 or access$5)same ((C or procedure or non$1object or legacy) near4 (C++ or object))) and 717/.ccls.) and object near4 map$4	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 14:48
-	94	(((map$3 or link$4 or inherit$5 or refer$5 or retriev$5 or access$5)same ((C or procedure or non$1object or legacy) near4 (C++ or object))) and 717/.ccls.) and cast$3	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 14:54
-	1	(((map$3 or link$4 or inherit$5 or refer$5 or retriev$5 or access$5)same ((C or procedure or non$1object or legacy) near4 (C++ or object))) and 717/.ccls.) and (cast$3 same (encapsulat$5 or abstract$5) near4 (C or non$1object))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 14:55
-	2	(((map$3 or link$4 or inherit$5 or refer$5 or retriev$5 or access$5)same ((C or procedure or non$1object or legacy) near4 (C++ or object))) and 717/.ccls.) and (cast$3 same (encapsulat$5 or abstract$5) same (C or non$1object))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 14:55

-	10	(((map$3 or link$4 or inherit$5 or refer$5 or retriev$5 or access$5)same ((C or procedure or non$1object or legacy) near4 (C++ or object))) and 717/.ccls.) and (cast$3 same (encapsulat$5 or abstract$5))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 14:55
-	2326	(map$3 or link$4 or inherit$5 or refer$5 or retriev$5 or access$5)same ((C) near4 (C++))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 15:03
-	72	((map$3 or link$4 or inherit$5 or refer$5 or retriev$5 or access$5)same ((C) near4 (C++))) and (map$3 or inherit$5 or retriev$5 or access$5)near5 (C) near4 (C++)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 15:14
-	565	(map$3 or link$4 or inherit$5 or retriev$5 or access$5) near3 (C or non$1object) near4 (C++ or (object or object$1oriented))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 15:33
-	494	(map$3 or link$4 or inherit$5 or retriev$5 or access$5) near3 (C or non$1object) near3 (C++ or (object or object$1oriented))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 15:16
-	494	((map$3 or link$4 or inherit$5 or retriev$5 or access$5) near3 (C or non$1object) near4 (C++ or (object or object$1oriented))) and (map$3 or link$4 or inherit$5 or retriev$5 or access$5) near3 (C or non$1object) near3 (C++ or (object or object$1oriented))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 15:16
-	35	(((map$3 or link$4 or inherit$5 or retriev$5 or access$5) near3 (C or non$1object) near4 (C++ or (object or object$1oriented))) and (map$3 or link$4 or inherit$5 or retriev$5 or access$5) near3 (C or non$1object) near3 (C++ or (object or object$1oriented))) and (encapsulat$5 and abstract$5 and cast$4)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 15:17
-	600	(map$3 or link$4 or inherit$5 or retriev$5 or access$5) near3 (C or non$1object or legacy) near4 (C++ or (object or object$1oriented))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 17:04
-	1	((map$3 or link$4 or inherit$5 or retriev$5 or access$5) near3 (C or non$1object or legacy) near4 (C++ or (object or object$1oriented))) and (overl$5 near4 memory)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 16:55
-	72	((map$3 or link$4 or inherit$5 or retriev$5 or access$5) near3 (C or non$1object or legacy) near4 (C++ or (object or object$1oriented))) and (over$7 or cover$4 or plac$4) near4 memory	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 15:39
-	93	((map$3 or link$4 or inherit$5 or retriev$5 or access$5) near3 (C or non$1object or legacy) near4 (C++ or (object or object$1oriented))) and (inherit$5 and encapsulat$5 and abstract$4)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 15:50
-	13	((map$3 or link$4 or inherit$5 or retriev$5 or access$5) near3 (C or non$1object or legacy) near4 (C++ or (object or object$1oriented))) and ((overl$5 or replac$4 or overwrit$4 or cover$3) near4 memory)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 16:56

-	7290	object near2 map$4	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 17:04
-	0	(object near2 map$4) and (object near3 map$4) same inherit$3 same (overl$3 and memory)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 16:59
-	0	(object near2 map$4) and (object near3 map$4) same inherit$3 same (overl$6 and memory)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 16:59
-	0	(object near2 map$4) and (object near3 map$4) same inherit$3 same (overl$9 and memory)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 17:00
-	66	(object near2 map$4) and (object near3 map$4) same inherit$3	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 17:00
-	78	(map$3 or link$4 or inherit$5 or retriev$5 or access$5) near3 (non$1object or legacy) near4 (C++ or (object or object$1oriented))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 19:46
-	1	((map$3 or link$4 or inherit$5 or retriev$5 or access$5) near3 (non$1object or legacy) near4 (C++ or (object or object$1oriented))) and inherit$4 same (non$1object)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 19:38
-	0	20020116700.URPN.	USPAT	2004/02/17 19:38
-	0	20020116700.URPN.	USPAT	2004/02/17 19:38
-	0	20020116700.URPN.	USPAT	2004/02/17 19:38
-	98	encapsulat$4 near3 (non$1object or legacy)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 19:46
-	138	(encapsulat$4 or inherit$4) near3 (non$1object or legacy)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 19:48
-	18	((encapsulat$4 or inherit$4) near3 (non$1object or legacy)) and (overl$4 or cover$4 or replac$4 or overwrit$4 or no or non or zero) near3 memory	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 19:56
-	25	(wrap$4 same (C or procedure or non$1object or legacy) near4 (C++ or object)) and 717/.ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 20:02
-	0	((wrap$4 same (C or procedure or non$1object or legacy) near4 (C++ or object)) and 717/.ccls.) and inheirt$4	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 20:02
-	15	((wrap$4 same (C or procedure or non$1object or legacy) near4 (C++ or object)) and 717/.ccls.) and inherit$4	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/18 10:22

-	0	C++ same wrap$4 same inherit$4 and type near2 cast$3	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 20:10
-	0	(C++ same wrap$4 same inherit$4) and (type near2 cast$3)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 20:11
-	3	(C++ same wrap$4 and inherit$4) and (type near2 cast$3)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 20:12
-	1	((C++ and C) same wrap$4 and inherit$4) and (type near2 cast$3)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 20:12
-	50	(C++ same wrap$4 and inherit$4) and (cast$3)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 20:13

> home > about > feedback > login
US Patent & Trademark Office

Search Results

Search Results for: **[map* <near> (legacy or non-object)<AND>((map*<AND>(((non-object or legacy) <near> inherit*<AND>((C++<AND>(((map* or link* or access* or retriev* or inherit*) <near> (non-object or legacy) and inherit*))))))))]**
Found **163** of **127,132 searched.**

Search within Results

> Advanced Search
> Search Help/Tips

Sort by: Title Publication Publication Date **Score** Binder

Results 1 - 20 of 163 short listing

Prev Page 1 2 3 4 5 6 7 8 9 Next Page

1 Towards nanocomputer architecture 90%
Paul Beckett , Andrew Jennings
Australian Computer Science Communications , Proceedings of the seventh Asia-Pacific conference on Computer systems architecture - Volume 6 January 2002
Volume 24 Issue 3

At the nanometer scale, the focus of micro-architecture will move from processing to communication. Most general computer architectures to date have been based on a "stored program" paradigm that differentiates between memory and processing and relies on communication over busses and other (relatively) long distance mechanisms. Nanometer-scale electronics --- nanoelectronics - promises to fundamentally change the ground-rules. Processing will be cheap and plentiful, interconnection expensive but ...

2 SmartFiles: an OO approach to data file interoperability 89%
Matthew Haines , Piyush Mehrotra , John Van Rosendale
ACM SIGPLAN Notices , Proceedings of the tenth annual conference on Object-oriented programming systems, languages, and applications October 1995
Volume 30 Issue 10

Data files for scientific and engineering codes typically consist of a series of raw data values whose description is buried in the programs that interact with these files. In this situation, making even minor changes in the file structure or sharing files between programs (interoperability) can only be done after careful examination of the data files and the I/O statements of the programs interacting with this file. In short, scientific data files lack self-description, and other self-describin ...

3 Customizing IDL mappings and ORB protocols 79%
Girish Welling , Maximilian Ott
IFIP/ACM Internati nal C nference n Distributed systems platf rms April 2000

Current mappings of IDL to implementation languages such as C++ or Java use CORBA specific data-types, which makes it imperative for an object implementation to be CORBA-compliant. While being completely CORBA-compliant ensures portability *and* interoperability, several classes of enterprise applications may *only* require interoperability with other CORBA applications. Other applications may be constrained by such factors as a large existing code-base or a widely used communicatio ...

4 The basic object system: supporting a spectrum from prototypes to hardened code 78%
Allen Dutoit , Sean Levy , Douglas Cunningham , Robert Patrick
ACM SIGPLAN Notices , Proceedings of the 11th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications October 1996

Volume 31 Issue 10
BOS is a prototype-based, object-oriented toolkit aimed at better supporting evolutionary software development. BOS attempts to support a spectrum of activities in one environment---ranging from rapid prototyping to code hardening. Features enabling rapid prototyping include a prototype-based object model, an interpreted language, run-time argument constraints, position and keyword arguments, and a user interface toolkit. BOS also provides features for code hardening such as multi-methods, multi ...

5 Access control with IBM Tivoli access manager 76%
Günter Karjoth
ACM Transactions on Information and System Security (TISSEC) May 2003
Volume 6 Issue 2
Web presence has become a key consideration for the majority of companies and other organizations. Besides being an essential information delivery tool, the Web is increasingly being regarded as an extension of the organization itself, directly integrated with its operating processes. As this transformation takes place, security grows in importance. IBM Tivoli Access Manager offers a shared infrastructure for authentication and access management, technologies that have begun to emerge in the com ...

6 Documentation: Documenting-in-the-large vs. documenting-in-the-small 74%
Scott R. Tilley
Proceedings of the 1993 conference of the Centre for Advanced Studies on Collaborative research: distributed computing - Volume 2 October 1993
There is a significant difference between documenting large programs and documenting small ones. By large programs we mean on the order of 1,000,000 lines, usually written by many different people over a long period of time. Most software documentation may be termed *documenting-in-the-small*, since it typically describes the program at the algorithm and dat a structure level. To understand large legacy systems, one needs *documenting-in-the-large*: documentation describing the high-le ...

7 Generic fuzzy reasoning nets as a basis for reverse engineering relational database applications 69%
Jens H. Jahnke , Wilhelm Schäfer , Albert Zündorf

ACM SIGSOFT S ftware Engineering N tes , Pr ceedings of the 6th European c nference held j intly with the 5th ACM SIGSOFT internati nal symp sium n F undati ns f s ftware engineering November 1997
Volume 22 Issue 6

8 The program understanding problem: analysis and a heuristic approach 68%
Steven Woods , Qiang Yang
Proceedings of the 18th international conference on Software engineering May 1996

Program understanding is the process of making sense of a complex source code. This process has been considered as computationally difficult and conceptually complex. So far no formal complexity results have been presented, and conceptual models differ from one researcher to the next. We formally prove that program understanding is NP hard. Furthermore, we show that even a much simpler subproblem remains NP hard. However we do not despair by this result, but rather offer an attractive problem so ...

9 Design and validation of QoS aware mobile internet access procedures for heterogeneous networks 64%
Giuseppe Bianchi , Nicola Blefari-Melazzi , Pauline M. L. Chan , Matthias Holzbock , Y. Fun Hu , Axel Jahn , Ray E. Sheriff
Mobile Networks and Applications February 2003
Volume 8 Issue 1

In this paper, the requirements for personal environments mobility are addressed from terminal and network perspectives. Practical mobility and Quality of Service (QoS) aware solutions are proposed for a heterogeneous network, comprising of satellite and terrestrial access networks connected to an IP core network. The aim, in adopting a heterogeneous environment, is to provide global, seamless service coverage to a specific area, allowing access to services independently of location. An importan ...

10 Routing: BANANAS: an evolutionary framework for explicit and multipath routing in the internet 63%
H. Tahilramani Kaur , S. Kalyanaraman , A. Weiss , S. Kanwar , A. Gandhi
Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture August 2003

Today the Internet offers a single path between end-systems even though it intrinsically has a large multiplicity of paths. This paper proposes an evolutionary architectural framework "BANANAS" aimed at simplifying the introduction of multipath routing in the Internet. The framework starts with the observation that a path can be encoded as a short hash ("PathID") of a sequence of globally known identifiers. The PathID therefore has global significance (unlike MPLS or ATM labels). This property a ...

11 Code migration through transformations: an experience report 63%
K. Kontogiannis , J. Martin , K. Wong , R. Gregory , H. Müller , J. Mylopoulos
Proceedings of the 1998 conference of the Centre for Advanced Studies on Collaborative research November 1998

One approach to dealing with spiraling maintenance costs, manpower shortages and frequent breakdowns for legacy code is to "migrate" the code into a new platform and/or programming language. The objective of this paper is to explore the feasibility of semiautomating such a migration process in the presence of performance and other constraints for the migrant code. In particular, the paper reports on an experiment involving a medium-size software system written in PL/IX. Several modules of the sy ...

12 Distributed programming with intermediate IDL 62%
Gary W. Smith , Richard A. Volz
ACM SIGAda Ada Letters , Proceedings f the ninth internati nal w rksh p n Real-time Ada June 1999
Volume XIX Issue 2

Several heterogeneous-language distributed programming systems have been developed which either use an explicit Interface Definition Language (IDL) for the specification of distributed objects or which directly translate server language specifications to corresponding client language representations. In this paper, we present a new approach which combines the advantages of these prior systems. Our approach uses an IDL as an implicit intermediate step in the translation from server to client lang ...

13 Migration of procedural systems to network-centric platforms 57%
Prashant Patil , Ying Zou , Kostas Kontogiannis , John Mylopoulos
Proceedings of the 1999 conference of the Centre for Advanced Studies on Collaborative research November 1999

Technologies developed over the past few years such as CORBA, Java and the Web, have made it easier to build and deploy distributed object applications. These technologies have also made a visible impact on legacy software system evolution.This paper focuses on the methods for re-engineering procedural systems into new Network-Centric platforms. The first step of this re-engineering method is to migrate a legacy system into an object oriented architecture. The extraction of the object oriented a ...

14 The SPEEDES Persistence Framework and the Standard Simulation Architecture 57%
Dr. Jeffrey S. Steinman , Jennifer W. Wong
Proceedings of the seventeenth workshop on Parallel and distributed simulation June 2003

This paper provides an overview of the SPEEDESpersistence framework that is currently used to automatecheckpoint/restart for the Joint Simulation System. Thepersistence framework interfaces are documented in thispaper and are proposed standards for the StandardSimulation Architecture.The persistence framework fundamentally keeps trackof memory allocations and pointer references within ahigh-speed internal database linked with applications.With persistence, an object, and the collection of object ...

15 Modeling coalition warfare: a multi-sided simulation design 56%
Kevin Brandt , Ellen Roland
Proceedings of the 25th conference on Winter simulation December 1993

16 Lessons from the battlefield 53%
Thomas P. Vayda
ACM SIGPLAN Notices , Proceedings of the tenth annual conference on Object-oriented programming systems, languages, and applications October 1995
Volume 30 Issue 10

The pragmatic aspects of deploying large scale Object Oriented (OO) applications are examined. The focus is on identifying some of the main obstacles that arise in typical large scale OO projects, and offering hints about effective solutions. This The topics are based on a number of actual large scale projects in which the author participated in a It significant capacity and solutions that he adopted or developed to deal with the

problems encountered.

17 Workshop on object-oriented legacy systems and software evolution 52%
Antero Taivalsaari , Roland Trauter , Eduardo Casais
ACM SIGPLAN OOPS Messenger , Addendum to the proceedings f the 10th annual conference on Object-oriented programming systems, languages, and applications (Addendum) October 1995
Volume 6 Issue 4

18 A first-class approach to genericity 50%
Eric Allen , Jonathan Bannet , Robert Cartwright
ACM SIGPLAN Notices , Proceedings of the 18th ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications October 2003
Volume 38 Issue 11

This paper describes how to add first-class generic types---including mixins---to strongly-typed OO languages with nominal subtyping such as Java and C#. A generic type system is "first-class" if generic types can appear in any context where conventional types can appear. In this context, a mixin is simply a generic class that extends one of its type parameters, *e.g.*, a class C<T> that extends T. Although mixins of this form are widely used in Cpp (via templates), they are clumsy an ...

19 Turning light bulbs into objects 49%
Bernd Bruegge , Truman Fenton , Tae Wook Kim , Ricardo Pravia , Aseem Sharma , Benedict Fernandes , Seongju Chang , Volker Hartkopf
Addendum to the 1997 ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (Addendum) January 1997

20 Standards for system-level design: practical reality or solution in search of a question? 48%
Christopher K. Lennard , Patrick Schaumont , Gjalt de Jong , Anssi Haverinen , Pete Hardee
Proceedings of the conference on Design, automation and test in Europe January 2000

Results 1 - 20 of 163 short listing

Prev Page 1 2 3 4 5 6 7 8 9 Next Page

Search Results

Search Results for: **[map* <near> (legacy or non-object)<AND>((map*<AND>(((non-object or legacy) <near> inherit*<AND>((C++<AND>(((map* or link* r access* or retriev* or inherit*) <near> (non-object or legacy) and inherit*))))))))]**
Found **163** of **127,132 searched.**

Search within Results

> Advanced Search
> Search Help/Tips

Sort by: Title Publication Publication Date **Score** Binder

Results 1 - 20 of 163 short listing

Prev Page 1 2 3 4 5 6 7 8 9 Next Page

1 Towards nanocomputer architecture 90%
Paul Beckett , Andrew Jennings
Australian Computer Science Communications , Proceedings of the seventh Asia-Pacific conference on Computer systems architecture - Volume 6 January 2002
Volume 24 Issue 3

At the nanometer scale, the focus of micro-architecture will move from processing to communication. Most general computer architectures to date have been based on a "stored program" paradigm that differentiates between memory and processing and relies on communication over busses and other (relatively) long distance mechanisms. Nanometer-scale electronics --- nanoelectronics - promises to fundamentally change the ground-rules. Processing will be cheap and plentiful, interconnection expensive but ...

2 SmartFiles: an OO approach to data file interoperability 89%
Matthew Haines , Piyush Mehrotra , John Van Rosendale
ACM SIGPLAN Notices , Proceedings of the tenth annual conference on Object-oriented programming systems, languages, and applications October 1995
Volume 30 Issue 10

Data files for scientific and engineering codes typically consist of a series of raw data values whose description is buried in the programs that interact with these files. In this situation, making even minor changes in the file structure or sharing files between programs (interoperability) can only be done after careful examination of the data files and the I/O statements of the programs interacting with this file. In short, scientific data files lack self-description, and other self-describin ...

3 Customizing IDL mappings and ORB protocols 79%

Girish Welling , Maximilian Ott

IFIP/ACM Internati nal C nference n Distributed systems platforms April 2000

Current mappings of IDL to implementation languages such as C++ or Java use CORBA specific data-types, which makes it imperative for an object implementation to be CORBA-compliant. While being completely CORBA-compliant ensures portability *and* interoperability, several classes of enterprise applications may *only* require interoperability with other CORBA applications. Other applications may be constrained by such factors as a large existing code-base or a widely used communicatio ...

4 The basic object system: supporting a spectrum from prototypes to hardened code 78%

Allen Dutoit , Sean Levy , Douglas Cunningham , Robert Patrick

ACM SIGPLAN Notices , Proceedings of the 11th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications October 1996

Volume 31 Issue 10

BOS is a prototype-based, object-oriented toolkit aimed at better supporting evolutionary software development. BOS attempts to support a spectrum of activities in one environment---ranging from rapid prototyping to code hardening. Features enabling rapid prototyping include a prototype-based object model, an interpreted language, run-time argument constraints, position and keyword arguments, and a user interface toolkit. BOS also provides features for code hardening such as multi-methods, multi ...

5 Access control with IBM Tivoli access manager 76%

Günter Karjoth

ACM Transactions on Information and System Security (TISSEC) May 2003

Volume 6 Issue 2

Web presence has become a key consideration for the majority of companies and other organizations. Besides being an essential information delivery tool, the Web is increasingly being regarded as an extension of the organization itself, directly integrated with its operating processes. As this transformation takes place, security grows in importance. IBM Tivoli Access Manager offers a shared infrastructure for authentication and access management, technologies that have begun to emerge in the com ...

6 Documentation: Documenting-in-the-large vs. documenting-in-the-small 74%

Scott R. Tilley

Proceedings of the 1993 conference of the Centre for Advanced Studies on Collaborative research: distributed computing - Volume 2 October 1993

There is a significant difference between documenting large programs and documenting small ones. By large programs we mean on the order of 1,000,000 lines, usually written by many different people over a long period of time. Most software documentation may be termed *documenting-in-the-small*, since it typically describes the program at the algorithm and dat a structure level. To understand large legacy systems, one needs *documenting-in-the-large*: documentation describing the high-le ...

7 Generic fuzzy reasoning nets as a basis for reverse engineering relational database applications 69%

Jens H. Jahnke , Wilhelm Schäfer , Albert Zündorf

ACM SIGSOFT Software Engineering Notes , Pr ceedings f the 6th Eur pean c nference held j intly with the 5th ACM SIGSOFT internati nal symp sium n F undations f s ftware engineering November 1997
Volume 22 Issue 6

8 The program understanding problem: analysis and a heuristic approach 68%
Steven Woods , Qiang Yang
Proceedings of the 18th international conference on Software engineering May 1996

Program understanding is the process of making sense of a complex source code. This process has been considered as computationally difficult and conceptually complex. So far no formal complexity results have been presented, and conceptual models differ from one researcher to the next. We formally prove that program understanding is NP hard. Furthermore, we show that even a much simpler subproblem remains NP hard. However we do not despair by this result, but rather offer an attractive problem so ...

9 Design and validation of QoS aware mobile internet access procedures for heterogeneous networks 64%
Giuseppe Bianchi , Nicola Blefari-Melazzi , Pauline M. L. Chan , Matthias Holzbock , Y. Fun Hu , Axel Jahn , Ray E. Sheriff
Mobile Networks and Applications February 2003
Volume 8 Issue 1

In this paper, the requirements for personal environments mobility are addressed from terminal and network perspectives. Practical mobility and Quality of Service (QoS) aware solutions are proposed for a heterogeneous network, comprising of satellite and terrestrial access networks connected to an IP core network. The aim, in adopting a heterogeneous environment, is to provide global, seamless service coverage to a specific area, allowing access to services independently of location. An importan ...

10 Routing: BANANAS: an evolutionary framework for explicit and multipath routing in the internet 63%
H. Tahilramani Kaur , S. Kalyanaraman , A. Weiss , S. Kanwar , A. Gandhi
Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture August 2003

Today the Internet offers a single path between end-systems even though it intrinsically has a large multiplicity of paths. This paper proposes an evolutionary architectural framework "BANANAS" aimed at simplifying the introduction of multipath routing in the Internet. The framework starts with the observation that a path can be encoded as a short hash ("PathID") of a sequence of globally known identifiers. The PathID therefore has global significance (unlike MPLS or ATM labels). This property a ...

11 Code migration through transformations: an experience report 63%
K. Kontogiannis , J. Martin , K. Wong , R. Gregory , H. Müller , J. Mylopoulos
Proceedings of the 1998 conference of the Centre for Advanced Studies on Collaborative research November 1998

One approach to dealing with spiraling maintenance costs, manpower shortages and frequent breakdowns for legacy code is to "migrate" the code into a new platform and/or programming language. The objective of this paper is to explore the feasibility of semiautomating such a migration process in the presence of performance and other constraints for the migrant code. In particular, the paper reports on an experiment involving a medium-size software system written in PL/IX. Several modules of the sy ...

12 Distributed programming with intermediate IDL 62%
Gary W. Smith , Richard A. Volz
ACM SIGAda Ada Letters , Proceedings f the ninth international worksh p n Real-time Ada June 1999
Volume XIX Issue 2
Several heterogeneous-language distributed programming systems have been developed which either use an explicit Interface Definition Language (IDL) for the specification of distributed objects or which directly translate server language specifications to corresponding client language representations. In this paper, we present a new approach which combines the advantages of these prior systems. Our approach uses an IDL as an implicit intermediate step in the translation from server to client lang ...

13 Migration of procedural systems to network-centric platforms 57%
Prashant Patil , Ying Zou , Kostas Kontogiannis , John Mylopoulos
Proceedings of the 1999 conference of the Centre for Advanced Studies on Collaborative research November 1999
Technologies developed over the past few years such as CORBA, Java and the Web, have made it easier to build and deploy distributed object applications. These technologies have also made a visible impact on legacy software system evolution.This paper focuses on the methods for re-engineering procedural systems into new Network-Centric platforms. The first step of this re-engineering method is to migrate a legacy system into an object oriented architecture. The extraction of the object oriented a ...

14 The SPEEDES Persistence Framework and the Standard Simulation Architecture 57%
Dr. Jeffrey S. Steinman , Jennifer W. Wong
Proceedings of the seventeenth workshop on Parallel and distributed simulation June 2003
This paper provides an overview of the SPEEDESpersistence framework that is currently used to automatecheckpoint/restart for the Joint Simulation System. Thepersistence framework interfaces are documented in thispaper and are proposed standards for the StandardSimulation Architecture.The persistence framework fundamentally keeps trackof memory allocations and pointer references within ahigh-speed internal database linked with applications.With persistence, an object, and the collection of object ...

15 Modeling coalition warfare: a multi-sided simulation design 56%
Kevin Brandt , Ellen Roland
Proceedings of the 25th conference on Winter simulation December 1993

16 Lessons from the battlefield 53%
Thomas P. Vayda
ACM SIGPLAN Notices , Proceedings of the tenth annual conference on Object-oriented programming systems, languages, and applications October 1995
Volume 30 Issue 10
The pragmatic aspects of deploying large scale Object Oriented (OO) applications are examined. The focus is on identifying some of the main obstacles that arise in typical large scale OO projects, and offering hints about effective solutions. This The topics are based on a number of actual large scale projects in which the author participated in a It significant capacity and solutions that he adopted or developed to deal with the

problems encountered.

17 Workshop on object-oriented legacy systems and software evolution 52%
Antero Taivalsaari , Roland Trauter , Eduardo Casais
ACM SIGPLAN OOPS Messenger , Addendum t the pr ceedings f the 10th annual conference on Object-oriented programming systems, languages, and applications (Addendum) October 1995
Volume 6 Issue 4

18 A first-class approach to genericity 50%
Eric Allen , Jonathan Bannet , Robert Cartwright
ACM SIGPLAN Notices , Proceedings of the 18th ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications October 2003
Volume 38 Issue 11

This paper describes how to add first-class generic types---including mixins---to strongly-typed OO languages with nominal subtyping such as Java and C#. A generic type system is "first-class" if generic types can appear in any context where conventional types can appear. In this context, a mixin is simply a generic class that extends one of its type parameters, *e.g.*, a class C<T> that extends T. Although mixins of this form are widely used in Cpp (via templates), they are clumsy an ...

19 Turning light bulbs into objects 49%
Bernd Bruegge , Truman Fenton , Tae Wook Kim , Ricardo Pravia , Aseem Sharma , Benedict Fernandes , Seongju Chang , Volker Hartkopf
Addendum to the 1997 ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (Addendum) January 1997

20 Standards for system-level design: practical reality or solution in search of a question? 48%
Christopher K. Lennard , Patrick Schaumont , Gjalt de Jong , Anssi Haverinen , Pete Hardee
Proceedings of the conference on Design, automation and test in Europe January 2000

Results 1 - 20 of 163 short listing

Prev Page 1 2 3 4 5 6 7 8 9 Next Page

Your search matched **12** of **1003743** documents.
A maximum of **500** results are displayed, **15** to a page, sorted by **Relevance Descending** order.

Refine This Search:
You may refine your search by editing the current search expression or enter new one in the text box.

(map* or link* or access* or retriev*) <near> (non ob Search

Check to search within this result set

Results Key:
JNL = Journal or Magazine **CNF** = Conference **STD** = Standard

---

1 **Organized C: a unified method of handling data in CAD algorithms a databases**
Soukup, J.;
Design Automation Conference, 1990. Proceedings. 27th ACM/IEEE , 24-28 Ju 1990
Pages:425 - 430

[Abstract] [PDF Full-Text (520 KB)] **IEEE CNF**

---

2 **Reengineering relational databases to object-oriented: constructing class hierarchy and migrating the data**
Alhajj, R.; Polat, F.;
Reverse Engineering, 2001. Proceedings. Eighth Working Conference on , 2-5 2001
Pages:335 - 344

[Abstract] [PDF Full-Text (808 KB)] **IEEE CNF**

---

3 **Capsules: a shared memory access mechanism for Concurrent C/C+**
Gehani, N.H.;
Parallel and Distributed Systems, IEEE Transactions on , Volume: 4 , Issue: 7 1993
Pages:795 - 811

[Abstract] [PDF Full-Text (1260 KB)] **IEEE JNL**

---

4 **The role of ACTIVEX and COM in ATE**
Fertitta, K.G.; Harvey, J.M.;
AUTOTESTCON '99. IEEE Systems Readiness Technology Conference, 1999. IEEE , 30 Aug.-2 Sept. 1999

Pages:35 - 51

[Abstract] [PDF Full-Text (920 KB)] **IEEE CNF**

---

5 **M bility management in third-generation all-IP networks**
Chiussi, F.M.; Khotimsky, D.A.; Krishnan, S.;
Communications Magazine, IEEE , Volume: 40 , Issue: 9 , Sep 2002
Pages:124 - 135

[Abstract] [PDF Full-Text (1948 KB)] **IEEE JNL**

---

6 **The WebUmbrella Web-based access to distributed plasma-physics measurement data**
Niderost, B.; van de Giessen, M.; Lourens, W.; Krom, J.;
Nuclear Science, IEEE Transactions on , Volume: 49 , Issue: 3 , June 2002
Pages:1579 - 1583

[Abstract] [PDF Full-Text (205 KB)] **IEEE JNL**

---

7 **An object-oriented programming methodology for a conventional programming environment**
Breen, D.E.; Getto, P.H.; Apodaca, A.A.;
Software Engineering, 1988 Software Engineering 88., Second IEE/BCS Conference: , 11-15 Jul 1988
Pages:65 - 72

[Abstract] [PDF Full-Text (660 KB)] **IEE CNF**

---

8 **Towards normalized iconic indexing**
Petraglia, G.; Sebillo, M.; Tucci, M.; Tortora, G.;
Visual Languages, 1993., Proceedings 1993 IEEE Symposium on , 24-27 Aug.

Pages:392 - 394

[Abstract] [PDF Full-Text (212 KB)] **IEEE CNF**

---

9 **S/sup 2/GA: a soft structured genetic algorithm, and its applicati n Web mining**
Nasaroui, O.; Dasgupta, D.; Pavuluri, M.;
Fuzzy Information Processing Society, 2002. Proceedings. NAFIPS. 2002 Ann Meeting of the North American , 27-29 June 2002
Pages:87 - 92

[Abstract] [PDF Full-Text (590 KB)] **IEEE CNF**

---

10 **Inheriting and modifying behavior**
Soundarajan, N.; Fridella, S.;
Technology of Object-Oriented Languages and Systems, 1997. TOOLS 23. Proceedings , 28 July-1 Aug. 1997
Pages:148 - 162

[Abstract] [PDF Full-Text (264 KB)] **IEEE CNF**

---

11 **Tech(xt)s [links between text and technology]**
Malmud, A.D.;
Multimedia, IEEE , Volume: 7 , Issue: 4 , Oct.-Dec. 2000
Pages:6 - 9

[Abstract] [PDF Full-Text (768 KB)] **IEEE JNL**

---

12 **Design of object-oriented simulations in C++**
Joines, J.A.; Roberts, S.D.;
Simulation Conference Proceedings, 1995. Winter , 3-6 Dec. 1995
Pages:82 - 89

[Abstract] [PDF Full-Text (676 KB)] **IEEE CNF**

Home | Log-out | Journals | Conference Proceedings | Standards | Search by Author | Basic Search | Advanced Search | Join IEEE | Web Account | New this week | OPAC Linking Information | Your Feedback | Technical Support | Email Alerting | No Robots Please | Release Notes | IEEE Online Publications | Help | FAQ| Terms | Back to Top

Cast Operations 6

This chapter explains the new cast operations: const and volatile casts, reinterpret cast, static and dynamic casts.

The emerging C++ standard defines new cast operations that provide finer control over casting than previous cast operations. The `dynamic_cast<>` operator provides a way to check the actual type of a pointer to a polymorphic class. Otherwise, the new casts all perform a subset of the casts allowed by the classic cast notation. For example, `const_cast<int*>v` could be written `(int*)v`. The new casts simply categorize the variety of operations available to express the programmer's intent more clearly and allow the compiler to better check the code.

Cast Operations Options

To enable recognition of the cast operators, use the option `-features=castop` , which is the default. To disable recognition of the cast operators, use the option `-features=no%castop` .

Const and Volatile Cast

The expression `const_cast<T>(v)` can be used to change the "const" or "volatile" qualifiers of pointers or references. `T` must be a pointer, reference, or pointer to member type. If *cv1* and *cv2* are some combination of `const` and `volatile` qualifiers (that is, *cv1* is `volatile` and *cv2* is `const volatile`), `const_cast` can convert a value of type "pointer to *cv1* `T`" to "pointer to *cv2* `T`", or "pointer to member of type *cv1* `T`" to "pointer to member of type *cv2* `T`". If we have an lvalue of type *cv1* T, then `const_cast` can convert it to "reference to type *cv2* `T`". (An lvalue names an object in such a way that its address can be taken.)

```
class A { public: virtual void f();
                  int i; };
extern const int A::* cimp;
extern const volatile int* cvip;
extern int* ip;
void use_of_const_cast( )
    { const A a1;
      const_cast<A&>(a1).f( );    // remove const
      a1.*(const_cast<int A::*> cimp) = 1;     // remove const
      ip = const_cast<int*> cvip; }    // remove const and volatile
```

Reinterpret Cast

The expression `reinterpret_cast<T>(v)` changes the interpretation of the value of the expression `v`. It will convert from pointer types to integers and back again, between two unrelated pointers, pointers to members, or pointers to functions. The only guarantee on such casts is that a cast to a new type, followed by a cast back to the original type, will have the original value. It is legal to cast an lvalue of type `T1` to type `T2&` if a pointer of type T1* can be converted to a pointer of type `T2*` by a `reinterpret_cast`. `reinterpret_cast` cannot be used to convert between pointers to two different classes that are related by inheritance (use `static_cast` or `dynamic_cast`), nor can it be used to cast away `const` (use `const_cast`).

```
class A { public: virtual void f( ); };
void use_of_reinterpret_cast( )
    { A a1;
      const A a2;
      int i = reinterpret_cast<int>(&a1);    // grab address
      const int j = reinterpret_cast<int>(&a2); }  // grab address
```

Static Cast

The expression `static_cast<T>(v)` converts the value of the expression v to that of type T. It can be used for any cast that is performed implicitly on assignment. In addition, any value may be cast to `void`,

and any implicit cast can be reversed if that cast would be legal as an old-style cast. It cannot be used to cast away `const`.

```
class B              { public: virtual void g( ); };
class C : public B { public: virtual void g( ); };
void use_of_static_cast( )
    { C c;
      // an explicit temporary lvalue to the base of c, a B
      B& br = c;
      br.g( );    // call B::g instead of C::g
      // a static_cast of an lvalue to the base of c, a B
      static_cast<B&>(c).g( ); }    // call B::g instead of C::g
```

Dynamic Cast

A pointer or reference to a class can actually point to any class publicly derived from that class. Occasionally, it may be desirable to obtain a pointer to the fully-derived class, or to some other base class for the object. The dynamic cast provides this facility.

The dynamic type cast will convert a pointer or reference to one class into a pointer or reference to another class. That second class must be the fully-derived class of the object, or a base class of the fully-derived class.

In the expression dynamic_cast<T>(v), v is the expression to be cast, and T is the type to which it should be cast. T must be a pointer or reference to a complete class type, or "pointer to cv void", where cv is [`const`][`volatile`]. In the case of pointer types, if the specified class is not a base of the fully derived class, the cast returns a null pointer. In the case of reference types, if the specified class is not a base of the fully derived class, the cast throws a bad_cast exception. For example, given the class definitions:

```
class A         { public: virtual void f( ); };
class B         { public: virtual void g( ); };
class AB :      public virtual A, private B { };
```

The following function will succeed.

```
void simple_dynamic_casts( )
    { AB  ab;
      B*  bp   = (B*)&ab;  // cast needed to break protection
      A*  ap   = &ab;      // public derivation, no cast needed
      AB& abr = dynamic_cast<AB&>(*bp);  // succeeds
      ap = dynamic_cast<A*>(bp);         assert( ap != NULL );
      bp = dynamic_cast<B*>(ap);         assert( bp == NULL );
      ap = dynamic_cast<A*>(&abr);       assert( ap != NULL );
      bp = dynamic_cast<B*>(&abr);       assert( bp == NULL ); }
```

In the presence of virtual inheritance and multiple inheritance of a single base class, the actual dynamic cast must be able to identify a unique match. If the match is not unique, the cast fails. For example, given the additional class definitions:

```
class AB_B :      public AB,        public B  { };
class AB_B__AB : public AB_B,       public AB { };
```

The following function will succeed:

```
void complex_dynamic_casts( )
    {
      AB_B__AB ab_b__ab;
      A*ap = &ab_b__ab;
                    // okay: finds unique A statically
      AB*abp = dynamic_cast<AB*>(ap);
                    // fails: ambiguous
      assert( abp == NULL );
             // STATIC ERROR: AB_B* ab_bp = (AB_B*)ap;
                    // not a dynamic cast
      AB_B*ab_bp = dynamic_cast<AB_B*>(ap);
                    // dynamic one is okay
      assert( ab_bp != NULL );
     }
```

The null-pointer error return of `dynamic_cast` is useful as a condition between two bodies of code, one to handle the cast if the type guess is correct, and one if it is not.

```
void using_dynamic_cast( A* ap )
    {
      if ( AB *abp = dynamic_cast<AB*>(ap) )
          { // abp is non-null,
            // so ap was a pointer to an AB object
            // go ahead and use abp
            process_AB( abp ); }
      else
          { // abp is null,
            // so ap was NOT a pointer to an AB object
            // do not use abp
            process_not_AB( ap );
     }
```

If run-time type information has been disabled, i.e. -features=no%rtti, (See Chapter 5, "RTTI"), the compiler converts dynamic_cast to static_cast and issues a warning.

If exceptions have been disabled (See Chapter 4, "Exception Handling"), the compiler converts dynamic_cast<T&> to static_cast<T&> and issues a warning. The dynamic cast to a reference may require an exception in normal circumstances.

Dynamic cast is necessarily slower than an appropriate design pattern, such as conversion by virtual functions. See *Design Patterns: Elements of Reusable Object-Oriented Software* by Erich Gamma et al.